by Dan Karmann (USA)

Build an MCU-Based Bicycle Computer

The EBikeMeter is a microcontroller-based bicycle computer that displays a variety of data. The design continuously stores the data on an on-board SD memory card and features a firmware file system that supports a PC-compatible, FAT-based file format.

A friend of mine recently upgraded his home-built recumbent bicycle with an electric-assist hub motor to ease his work commute. He asked me if I could design a monitoring/logging system and bicycle computer to display and help characterize the bicycle's operation. The result is the EBikeMeter, an Atmel ATmega328P-based bicycle computer (see Photo 1).

This article provides a system overview and high-level descriptions of the hardware and the operating firmware. It also includes information about choosing and modifying the SD card file system, assembling the EBikeMeter, mounting the computer on a bicycle, and configuring and operating the unit.

SYSTEM OVERVIEW

The EBikeMeter's system requirements include displaying and monitoring the real-time speed, trip duration, elapsed trip mileage, temperature, motor current draw, battery voltage, and power consumption on a backlit four-line by 20-character LCD. The system continuously logs all those items as well as the minimum battery voltage, the maximum current draw, the maximum wattage, the watt hours, the amp hours, the average trip speed, the maximum trip speed, and the odometer readings to a removable on-board SD memory card. The EBikeMeter's system also controls the LCD, monitors user push buttons for the user interface, and provides a control output for speed/current/voltage-controlled throttle override. Additionally, a serial interface is provided for system configuration and firmware upgrades.

Figure 1 shows a block diagram of the EBikeMeter. An Atmel ATmega328P microcontroller is at the heart of the system. To measure speed and distance traveled, it receives pulse inputs from the speed sensor, which is mounted on the bicycle's front fork. The ADC inputs to the ATmega328P are used to monitor the battery voltage and motor current. The system's user interface is via a backlit 4 × 20 LCD with temperature-compensated contrast and two push buttons. A standard SD memory card socket is connected to the ATmega328P's serial peripheral interface (SPI) bus. The throttle override signal is an ATmega328P PWM output.

Photo 1—The EBikeMeter is mounted on an electric-assisted recumbent bicycle's handlebar.

The system is powered from the 48-V battery associated with the electric motor and motor controller. I used a 5-V-only LCD and a 3-V-only SD card. Therefore, I used a 48-to-5-V converter along with a 3.3-V regulator to provide the required voltages. I used a reset supervisor to receive early notification of system shutdown so appropriate logging and EEPROM storage can be completed before power is lost.

THE HARDWARE

Figure 2 is the EBikeMeter's schematic. Most of the parts came from my electronics stockpile. The SD card socket was salvaged from an old digital camera. The ATmega328P (IC1) controls the EBikeMeter's operation and uses every pin. The internal 8-MHz internal RC oscillator is

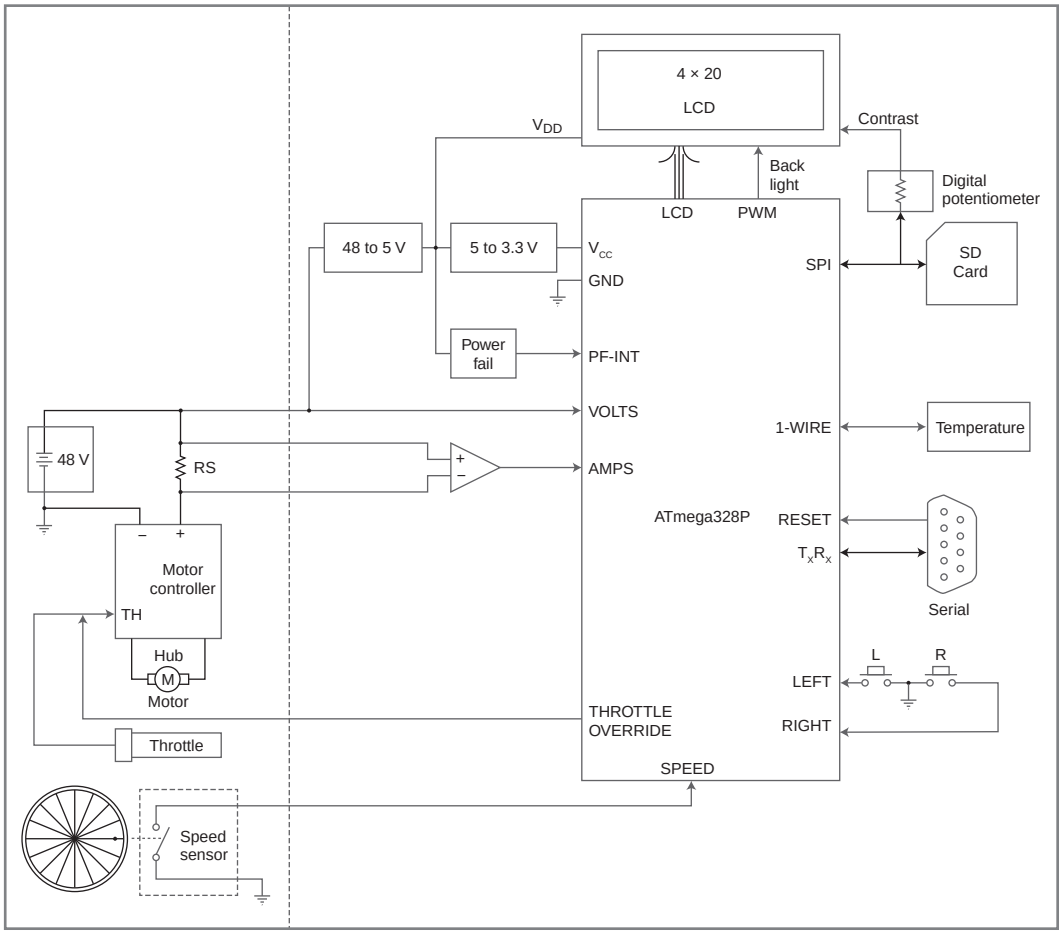


Figure 1—An Atmel ATmega328P microcontroller is at the heart of the EBikeMeter. The EBikeMeter's main components and the interfaces to the speed sensor, battery, motor controller, and throttle are also shown.

used, so no external crystal is required. The EBikeMeter operates at 3.3 V from the low dropout (LDO) regulator (IC4) from the 5-V power from the voltage converter (VM1) from the 48-V battery. The speed sensor input (from a magnetically activated reed switch or equivalent) is input to the ATmega328P's INT0 input. General-purpose input/output (GPIO) pins PC2 and PC3 monitor the user push-button switches SW1 (left) and SW2 (right). Other GPIO pins (PB6, PB7, and PD4-PD7) are used to interface to the LCD. The LCD backlight is controlled with the Timer2 PWM output using an external transistor Q1.

The ATmega328P SPI bus is used to interface to the SD memory card socket and the digital potentiometer (IC6), which controls the LCD contrast based on the current temperature. Note the heartbeat LED (on GPIO pin PB0) is also the SPI slave-select (SS) signal for the digital potentiometer's SPI bus. So, whenever SPI-based accesses are happening to the digital potentiometer or the SD card, the heartbeat LED must first be turned off! The temperature is obtained from a Maxim Integrated Products 1-Wire device (IC5) from another GPIO pin (PC5). GPIO pin PC4 monitors the power-fail input from a reset supervisor device (IC3) using pin-change interrupts for early power-down notification.

The scaled battery voltage is monitored on one of the ATmega328P's ADC channels (ADC1). The scaled voltage (from R1 and R2) seen by the processor is approximately $V_{BATT}/20$, which enables a battery voltage of up to 66 V to be monitored.

An external high-side, low-ohm shunt resistor (RS) is used to sense the electric drive motor current. It is then amplified and scaled by IC2 and filtered by R4, C3, and C5 before it is monitored on the ATmega328P's ADC 0 channel. With a 2.5-mΩ external shunt resistor, the current's voltage seen by the processor is approximately:

$$(2.5\ \text{m}\Omega \times \text{motor current} \times 20) + \left(\frac{3.3\ \text{V}}{2}\right)$$

This enables a motor current up to about 33 A to be monitored. Jeff Bachiochi wrote about using the MAX4081T (IC2) in this type of application in his article, "Electric Movement and Control" (*Circuit Cellar* 199, 2007). Note: The eight-pin MAX4081T device (IC2) is shown in the schematic superimposed on a six-pin DIP outline. This is because the MAX4081T is only available as a surface-mount device and is soldered to a six-pin DIP socket as a through-hole component. Also note: IC2 has two sources of V_{CC} power, one from the 5-V supply and one from the current shunt resistor's positive side. This was needed for initial testing when using an external 5-V supply without the 48-V battery. I noticed the IC2 REF1A pin was acting as a short circuit to ground without an IC2 V_{CC} voltage. By supplying both voltages via D10 and D11, if the 48-V battery voltage is ever lost (due to a blown fuse or broken connection), the 3.3-V voltage supply will not be affected and the rest of the circuit will continue operation.

The throttle override mechanism's intent is for the EBikeMeter to act as a governor to limit operation of the bicycle electric-assist motor to a maximum speed, maximum motor current, and/or minimum operating battery voltage. This is accomplished by overriding the throttle control signal from the throttle to the

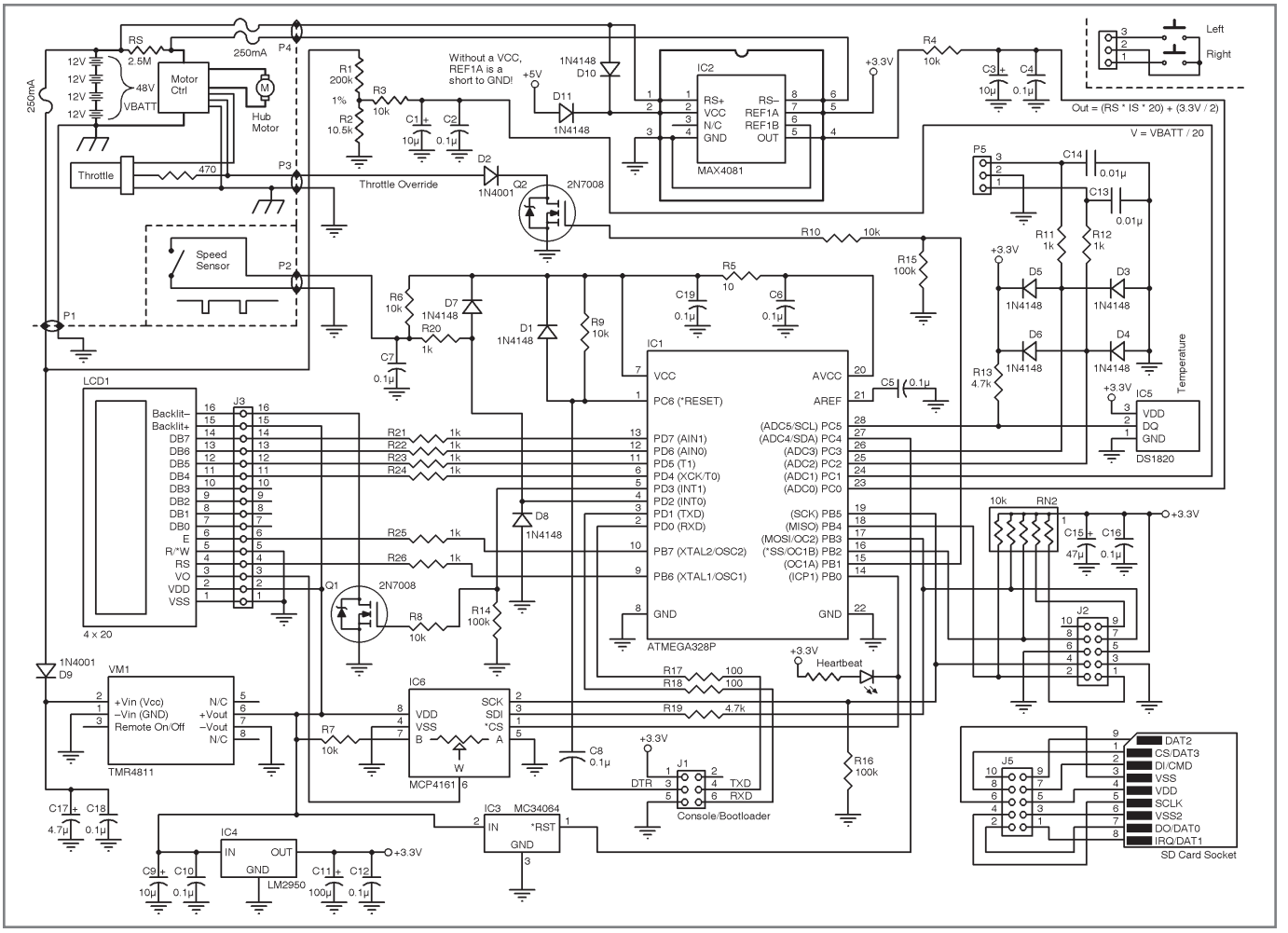


Figure 2—The complete EBikeMeter schematic depicts the off-board interfaces to the push-button switches, an SD card, a speed sensor, a battery, a motor controller, a throttle, and a serial console/bootloader interface.

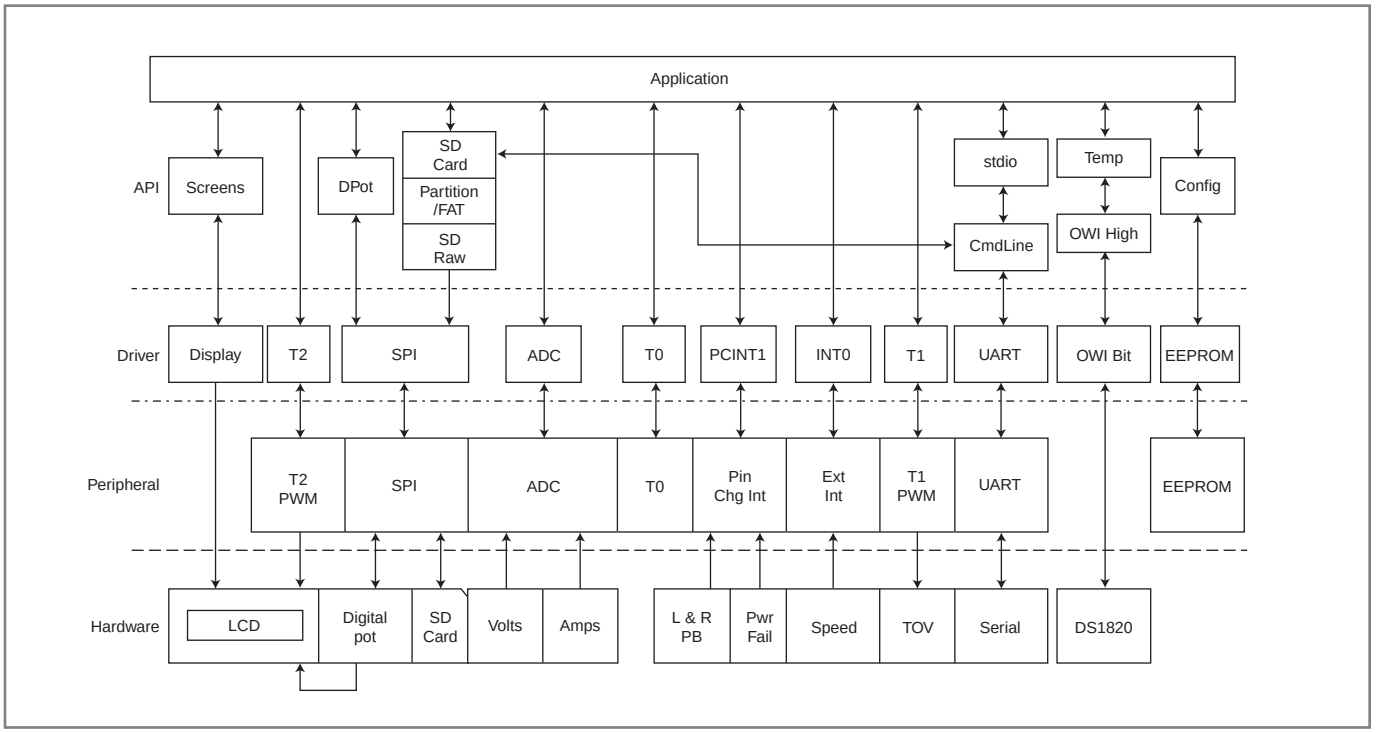


Figure 3—The EBikeMeter includes many firmware and hardware layers.

motor controller.

The typical control signal from the throttle control is a 1-to-4-V analog voltage, with the highest voltage indicating full throttle. The circuitry that implements the throttle override signal is a PWM/GPIO output from the processor to diode D2 via Q2, R10, and R15 along with a current-limiting resistor added to the connection between the throttle and the motor controller to protect the throttle's circuitry. When a set speed, current, or voltage limit is reached, the throttle override signal overrides the control signal from the throttle, preventing it from controlling the motor controller. This is a one-way override signal (i.e., it can only cause the motor controller to slow down, not speed up). If there is a speed control override, the bicycle speed should be limited to a certain maximum value with electric assistance.

Although the motor controller cannot increase the bicycle speed, the user can still pedal faster to increase the speed without electric motor assistance. If there is a motor current control override, the motor current draw should be limited to extend the battery's operating range, forcing the user to run slower or pedal harder. If there is a battery voltage override, the electric motor assist operation should be limited if the battery voltage is too low. This condition can cause battery failure.

"Although the motor controller cannot increase the bicycle speed, the user can still pedal faster to increase the speed without electric motor assistance. If there is a motor current control override, the motor current draw should be limited to extend the battery's operating range, forcing the user to run slower or pedal harder."

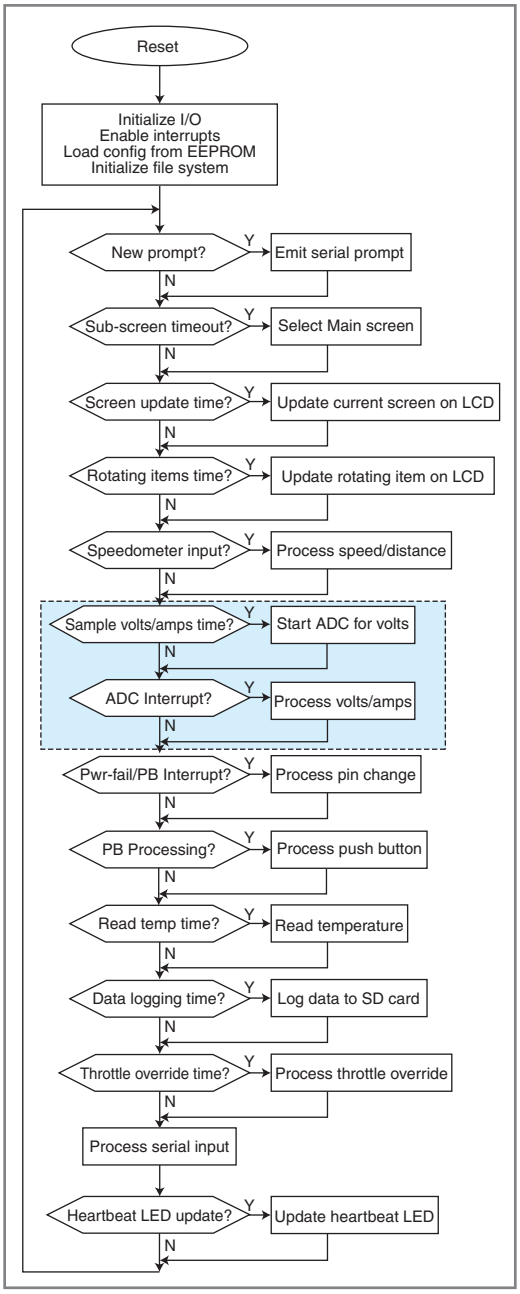


Figure 4—Here is the EBikeMeter firmware's start-up and main processing loop. An expanded view of the chart's highlighted section is available on *Circuit Cellar*'s FTP site.

The processor's UART is connected to an external serial interface to facilitate upgrading firmware and configuring the EBikeMeter configuration settings. To initiate the processor's on-board bootloader for firmware upgrades, the serial DTR signal is used to reset the processor. The EBikeMeter's serial interface's electrical interface is 3.3-V TTL level signals. To connect to a PC COM port, an external TTL-to-RS-232 converter is required (e.g., a MAX3232-style converter device). If it's needed, the 3.3-V power line is also supplied on the serial connector to power the external TTL-to-RS-232 converter. The required RS-232 signals needed are TxD, RxD, Gnd, and DTR. DTR is only needed for firmware upgrades via a MCS Electronics BootLoader, which I used because I first started programming AVRs using the BASCOM-AVR BASIC environment, which included the MCS BootLoader capability. I use the same firmware bootloader, even in the WinAVR C development environment, using a standalone host bootloader application. Additional schematic detail information can be found in section 7.2 of the EBikeMeter Reference Manual, which is available on *Circuit Cellar*'s FTP site.

THE FIRMWARE

The EBikeMeter firmware development was done in C using the Atmel AVR Studio 4 development environment with the WinAVR C compiler. The EBikeMeter firmware periodically monitors an electric bicycle's battery voltage, motor current, ambient temperature, and bicycle speed. Various measured and calculated statistics from this monitored data are displayed on a 4 × 20 backlit LCD and some of the monitored data items are maintained in EEPROM. The LCD is organized as five screens selectable with two momentary push buttons. For nighttime operation, a push button can be used to enable the display's LED backlight. When the bicycle is moving, various measured and calculated statistics are continuously logged on an SD card using a PC-compatible file system. Additionally, based on user-configured settings, the EBikeMeter firmware can limit the electric motor controller's throttle input to limit bicycle speed, motor current draw, and/or minimum operating battery voltage. An external serial interface is also provided for firmware upgrades and EBikeMeter configuration. Layered hardware/firmware architecture is used to structure the EBikeMeter firmware as a classic small embedded system with a foreground main loop and background interrupt processing (see Figure 3).

After reset, the various processor I/O modules are initialized, the configuration items are loaded from EEPROM, interrupts are enabled, and the main

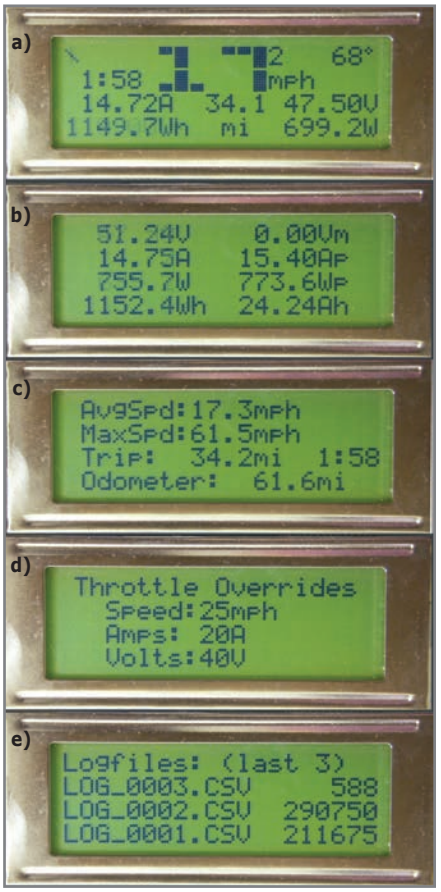


Photo 2—The EBikeMeter's LCD screens in sequence are: Main **(a)**, Power **(b)**, Speed/Distance **(c)**, Throttle Override **(d)**, and Recent Log Files List **(e)**.

loop is entered, as shown in Figure 4. Interrupts are used to monitor the speedometer input, power-fail input, user push-button input, general timer-based event timing, UART transmit and receive operation, and ADC voltage and current input monitoring.

The EBikeMeter configuration and nonvolatile statistics data loaded from the processor's EEPROM at start-up include: speedometer wheel size, LCD backlight timeout, LCD subscreen timeout, LCD backlight brightness, LCD contrast correction minimum temperature and digital potentiometer adjustment, speed throttle override speed, motor current throttle override current, and minimum battery throttle override voltage. The nonvolatile statistics items include: minimum operating voltage, peak motor current, amp hours, peak watts, watt hours, maximum trip speed, trip distance, trip duration, and odometer mileage. The current log file number is also retrieved from EEPROM.

When initializing the on-board file system, the SD card's first partition is opened and checked for supported partition type and the file allocation table (FAT) system is opened and checked for supported FAT type. Because of the ATmega328P's limited code space, the EBikeMeter's file system is restricted to a FAT-12 or FAT-16 file system. Newer FAT-32 (and SDHC card) file systems are not supported. Due to the limited amount of files that can be stored in a FAT-based file system's root directory, all EBikeMeter log files are stored in the "LOGS" subdirectory.

There are five available screens (see Photo 2). Each time through the main loop, the LCD screen timer is checked to see if the currently displayed screen needs updating (using customized "BIG numbers" for speed on the Main screen), if the Main screen needs to be redisplayed, and if the rotating items on the Main screen need to be changed. (More information about using "BIG numbers" can be found at the Arduino Forum, see the Resources section.) The speedometer input value is checked and the bicycle speed, trip duration, maximum speed, and average speed are calculated. Also, voltage and current values are accumulated and watts, watt hours, and amp hours are calculated. The power-fail input and user push buttons are checked and acted upon. If a power-fail event occurs, all nonvolatile items stored in EEPROM are updated and further processing stops until power is restored. User push-button inputs are run through a finite state machine (FSM) to debounce the push-button signals and determine which event action should be taken. Three types of push-button events are recognized: single-button pressed, dual-button pressed, and single-button hold.

The temperature is periodically read and a digital potentiometer is used to adjust the LCD contrast based on this value. The accumulated statistics data is continuously logged to the SD memory card's log file, but only if the bicycle is moving. All logging is suspended when the bicycle stops. Another periodic activity is determining if the throttle-limiting control needs to be changed based on the current speed, motor current, or battery voltage value and the associated configured limits. Finally, the serial interface is checked to determine if any commands have been received for processing. These commands are used for the various EBikeMeter configuration settings and basic SD card file management.

Additional firmware detail information, including application programming interfaces (APIs), device driver access, and source file organization can be found in section 8 of the EBikeMeter Reference Manual with the manual and the full source code, which are available on *Circuit Cellar*'s FTP site.

FILE SYSTEM SELECTION/MODIFICATION

The EBikeMeter firmware file system needs to support a PC-compatible FAT-based file format. I found several FAT-type file system firmware packages online, but most were too large to fit in the available flash memory space along with the rest of the application code. Some firmware packages were too large to fit even without any application code. I found a Roland Riegel MMC/SD/SDHC card library that was small enough to do the job, but it had one drawback: it only supported FAT-16 for SD cards from 32 MB to 2 GB. I had a surplus of FAT-12-based 8- and 16-MB SD cards I preferred to use in this application. After studying the Roland Riegel MMC/SD/SDHC card library code, I found I could add FAT-12 support to it for these smaller SD cards and it would still fit into the available flash memory (barely!) with the application code.

"The EBikeMeter configuration and nonvolatile statistics data loaded from the processor's EEPROM at start-up include: speedometer wheel size, LCD backlight timeout, LCD subscreen timeout, LCD backlight brightness, LCD contrast correction minimum temperature and digital potentiometer adjustment, speed throttle override speed, motor current throttle override current, and minimum battery throttle override voltage."

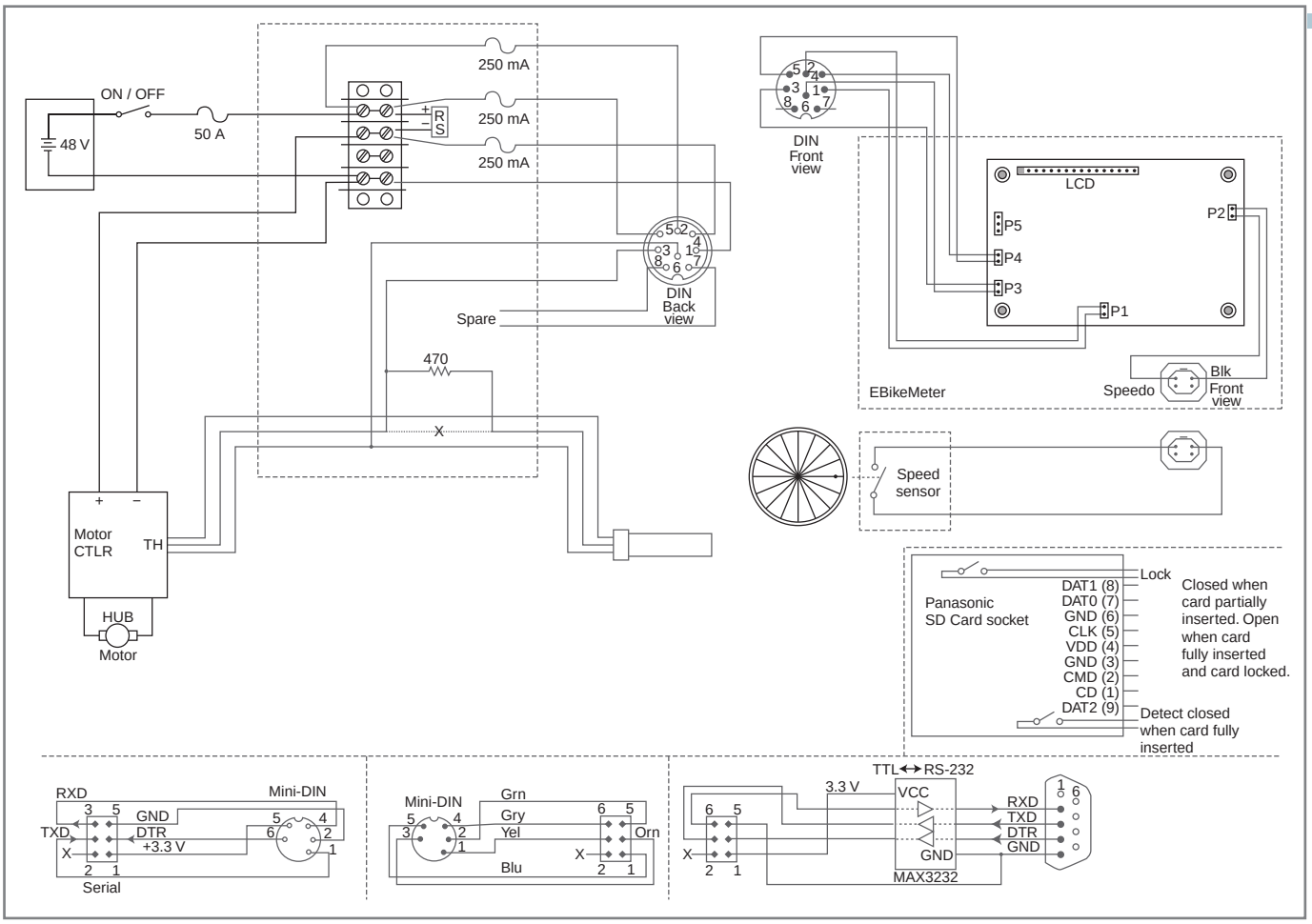


Figure 5—This detailed wiring diagram shows the connectors, speed sensor, voltage supply, current shunt, associated protective fuses, and throttle-control modifications. I also included a schematic of the Panasonic SD card, connections, and cables used for the serial interface adapter.

EBIKEMETER ASSEMBLY

The EBikeMeter circuitry was assembled on a modified Protostack PB-MC-AVR28 28-pin AVR full-size development board, which was sandwiched with the LCD board. This was placed in a small project box with the push buttons mounted on the left and right sides. The box had a custom cutout and window on its front for the LCD and featured a machined aluminum rear plate where the SD card socket and I/O connections for power, speedometer, serial port, and mounting brackets for the bicycle handlebar cross member were made. I used the FreePCB open-source PCB editor to help optimize the component layout for the point-to-point wiring. This tool was also used to design a couple of proposed layouts for PCBs, one for the LCD mounted on the PCB's component side and one for the LCD mounted on the PCB's noncomponent side, but neither have been fabricated into an actual board. More details (e.g., photos) of this assembly and layout are available at my *DLK Engineering* website (http://dlkeng.cwahi.net/EBikeMeter.htm).

INSTALLATION, CONFIGURATION, & OPERATION

Figure 5 shows how the EBikeMeter is wired with the bicycle's electric-assist motor controller, battery, throttle, and the speedometer sensor. The EBikeMeter was mounted on the handlebar cross brace and the speedometer sensor was attached to the front fork with a companion magnet attached to a front wheel spoke. The speedometer's wiring was routed from the front fork to the handlebar-mounted EBikeMeter. The power and throttle control signal wires were routed from the handlebar-mounted EBikeMeter along the bicycle frame, under the rider's seat to behind the rider, where a small box was placed that contained the voltage, current, and throttle control connections to the battery and motor controller.

Before using the EBikeMeter, several items need to be configured to the bicycle on which it is installed. This configuration can be accomplished using a serial connection to a PC at 19,200 bps using your favorite terminal emulation software. As I previously stated in the Hardware section, the EBikeMeter's serial interface is *not* RS-232, but is typically called a TTL interface. It requires an RS-232-to-TTL converter, or possibly a USB-to-TTL converter, such as those from FTDI. (Note that most FTDI converters do not supply the DTR signal, so they will work for configuration but not firmware upgrade.)

The most important configuration item is the wheel's circumference on which the speedometer sensor is installed. This controls the accuracy of the speed, trip distance, and odometer readings. Other configuration settings include: the initial odometer setting, the LCD backlight timeout and brightness, the LCD contrast adjustment base temperature value, the LCD

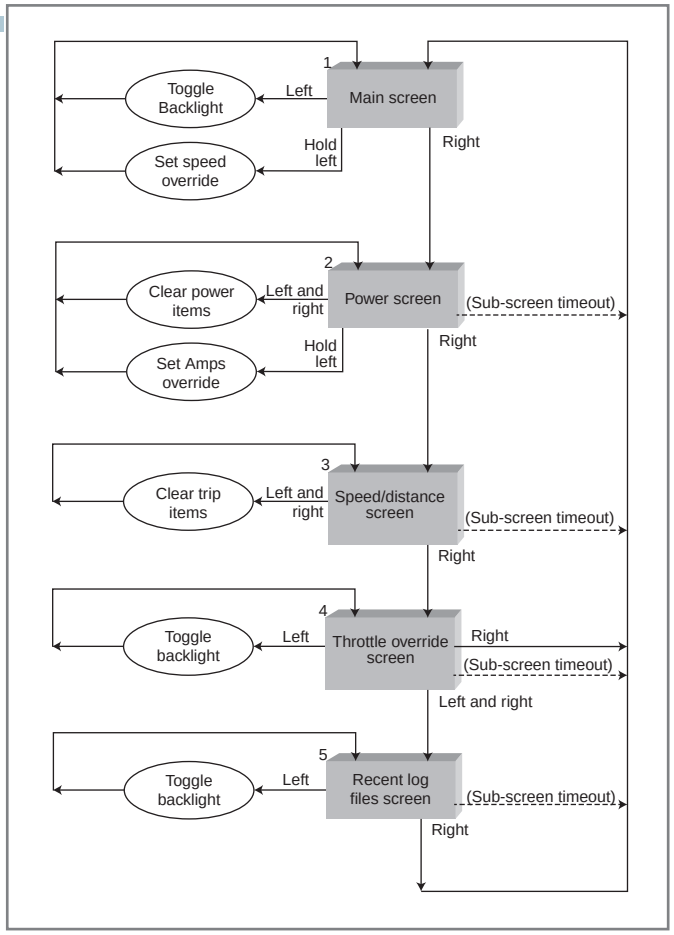


Figure 6—This diagram shows how the user push button EBikeMeter LCD screens navigation and function selection works.

subscreen timeout, and the throttle override control values. The LCD backlight is normally off. One of the push buttons can be used to turn it on. When the LCD backlight is turned on and the bicycle is moving, it will stay on, but when the bicycle stops moving, the LCD backlight will turn off after a configured timeout period. The LCD backlight brightness is adjustable for nighttime operation and should be unnecessary for daytime operation. The LCD contrast is temperature sensitive and requires temperature-based adjustment to remain visible at various temperatures. The configuration setting sets the temperature at which the adjustment begins. At every 0.5°C above this temperature, the on-board digital potentiometer is adjusted one step to change the contrast setting. Of the five LCD screens (shown in Photo 2), only the Main screen (with the speedometer display) can be indefinitely displayed, all the other screens have a timeout after which the display reverts back to the Main screen. The throttle override configuration control values limit the throttle for the maximum speed, the maximum motor current draw, and the minimum battery voltage.

The EBikeMeter operation is fairly simple. When it is powered on, a sign-on screen is shown for a few seconds then the Main screen appears. As shown in Figure 6, the right user push button is primarily used to navigate to other subscreens (see Photo 2). The Main screen is the primary screen needed as a bicycle computer, but pressing the right push button will select the next screen in the sequence: Main, Power, Speed/Distance, Throttle Override, and (optionally) Recent Log Files List. When on the Main screen, the Throttle Override screen, or the Recent Log Files List screen, the left user push button toggles the LCD backlight's state. When on the Power screen, the Speed/Distance screen, or the Throttle Override screen, the left and right user push buttons control clearing power statistics, clearing the trip statistics, and modifying the throttle override settings.

Additional configuration and operating information can be found in the EBikeMeter User's Manual and the EBikeMeter Reference Manual, which are available on *Circuit Cellar*'s FTP site.

TIME TO RIDE

This has been an interesting project and I have learned some new things. SD card interfacing and FAT file systems are no longer mysterious. Unfortunately, the biggest hurdle with this project was the lack of the AVR's available code space. It is completely full! This prevented me from making the code as robust as I would have liked, as many error conditions are not handled as well as they could be. I started with the ATmega168 (16-KB flash memory) and quickly moved to the ATmega328P (32-KB flash memory) once I realized how much code a FAT file system uses. I could have easily used a nonexistent larger drop-in replacement AVR part (e.g., ATmega648 with 64-KB flash memory). For those interested, the source code available on *Circuit Cellar*'s FTP site contains additional development and debugging commands from the serial interface. These can be used by turning some of the commands on and turning some of the other functionality off to fit into flash memory (SD card support and temperature support are two ideal candidates) from the defines.h header file. Additional information about this project is available on my EBikeMeter website.

Dan Karmann (dlkeng@msn.com) is a semi-retired embedded systems developer. He spent 27 years at AT&T Bell Labs/American Bell/AT&T Information Systems/Lucent Technologies Bell Labs/Avaya Labs. Prior to that, Dan worked as a CB radio repair technician while earning a BSEET (magna cum laude) from the University of Nebraska, Omaha. Dan spent six years as an Electronic Warfare technician in the U.S. Navy. He enjoys reverse engineering and documenting electronic device hardware and firmware from the 1980s and 1990s. He is a charter subscriber who has every issue of Circuit Cellar magazine and has been reading Steve Ciarcia's "Circuit Cellar" articles since the late 1970s. Dan's website is at http://dlkeng.cwahi.net.

PROJECT FILES

To download the code, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2012/269.

RESOURCES

Arduino Forum, "BIG Numbers from a Little LCD," 2008, http://arduino.cc/forum/index.php/topic,7245.0.html.

Atmel Corp., "Studio Archive," www.atmel.com/tools/STUDIOARCHIVE.aspx.

———, "AVR103: Using the EEPROM Programming Modes," 2005, www.atmel.com/images/doc2578.pdf.

———, "AVR318: Dallas 1-Wire Master," 2004, www.atmel.com/images/doc2579.pdf.

J. Bachiochi, "Electric Movement and Control," *Circuit Cellar* 199, 2007.

ConhisMotor Technology Co., Ltd., "Standard Controller 48 V 1,000 W," www.conhismotor.com/ProductShow.asp?id=49.

FreePCB, www.freepcb.com.

Future Technology Devices International, Ltd. (FTDI), "USB TTL Serial Cables," www.ftdichip.com/Products/Cables/USBTTLSerial.htm.

Grin Technologies, "Homepage of the Cycle Analyst," www.ebikes.ca/drainbrain.shtml.

D. Karmann, DLK Engineering, http://dlkeng.cwahi.net/EBikeMeter.htm.

MCS Electronics, "MCS BootLoader," Application Note 143, www.mcselec.com/index.php?option=com_content&task=view&id=159&Itemid=57.

Roland Riegel, "MMC/SD/SDHC Card Library," www.roland-riegel.de/sd-reader/index.html.

SourceForge, WinAVR, http://sourceforge.net/projects/winavr.

SOURCES

ATmega328P Microcontroller
Atmel Corp. | www.atmel.com

OAR-3 Open-air sense resistors
IRC | www.irctt.com

DS18S20 1-Wire digital thermometer and MAX4080/MAX4081 amplifiers
Maxim Integrated Products, Inc. | www.maxim-ic.com

BootLoader Windows application
MCS Electronics |www.mcselec.com

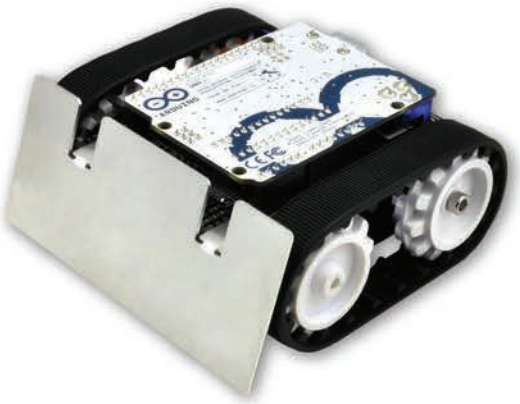
MCP414*x*/416*x*/424*x*/426*x* 7/8-Bit single/dual SPI digital potentiometer
Microchip Technology, Inc. | www.microchip.com

MC34064 Undervoltage sensing circuit
ON Semiconductor | www.onsemi.com

PB-MC-AVR28 28-Pin AVR development board
Protostack | www.protostack.com

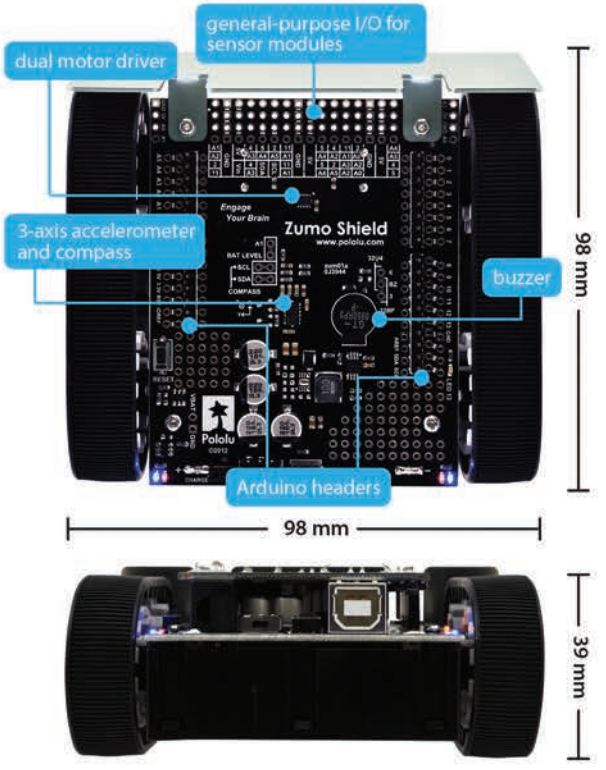
TMR 4811 DC/DC Converters
TRACO ELECTRONIC | www.tracopower.com

Get a little pushy.



Zumo

Small enough for Mini-Sumo; flexible enough to make it your own.



Put your Arduino on the right tracks with the new Zumo chassis and Arduino shield.

Pololu
Robotics & Electronics

Learn more at www.pololu.com/zumo